Fault-Tolerant Forwarding in the Face of Malicious Routers

Alper Tugay Mızrak, Keith Marzullo, Stefan Savage
University of California, San Diego
{amizrak, marzullo, savage}@cs.ucsd.edu

We are interested in a simple, yet increasingly important network security problem: how to detect the existence of compromised routers in a network and then remove them from the routing fabric.

The root of this problem arises from the key role that routers play in modern packet switched data networks. To a first approximation, networks can be modeled as a series of point-to-point links connecting pairs of routers to form a directed graph. Since few endpoints are directly connected, data must be forwarded - hop-by-hop - from router to router towards its destination. If a router is compromised, then it stands to reason that an attacker may drop, delay, reorder, corrupt, modify or divert any of the packets passing through. Such a capability can then be used to deny service to legitimate hosts, to implement ongoing network surveillance or to provide an efficient man-in-the-middle functionality for attacking end systems.

Such attacks are not mere theoretical curiosities, but they are actively employed in practice. Attackers have repeatedly demonstrated their ability to compromise routers, through combinations of social engineering and exploitation of weak passwords and latent software vulnerabilities [1, 7, 9]. One network operator recently documented over 5,000 compromised routers as well as an underground market for trading access to them [15].

Once a router is compromised an attacker need not modify the router's code base to exploit its capabilities. Current standard command-line interfaces from vendors such as Cisco and Juniper are sufficiently powerful to drop and delay packets, send copies of packets to a third party, or "divert" packets through a third party and back. In fact, several widely published documents provide a standard cookbook for transparently "tunneling" packets from a compromised router through an arbitrary third-party host and back again - effectively amplifying the attacker's abilities, including arbitrary packet sniffing, injection or modification [6, 14]. Such attacks can be extremely difficult to detect manually, and it can be even harder to isolate which particular router or group of routers has been compromised.

The problem of detecting and removing compromised routers can be thought of as an instance of anomalous behavior-based intrusion detection. That is, a compromised router can be identified by correct routers when it deviates from exhibiting expected behavior. This problem can be broken into three subproblems:

Traffic validation: Traffic information is the basis of detecting anomalous behavior: given traffic entering a part of the network, and an expected behavior of the routers in the network, anomalous behavior is detected when the monitored traffic leaving that part of the network differs significantly from what is expected. Implementing such validation involves tradeoffs between the overhead of monitoring, communication and accuracy.

Distributed detection: Any detection of a compromised router requires synchronizing the collection of traffic information and distributing the results. There are tradeoffs between the overhead of synchronization and information distribution with the precision of the detection.

Response: Once a router or set of routers are suspected of being faulty, some countermeasure needs to be taken. A suspected router can be complexly isolated or only removed from paths whose traffic it is modifying.

There are two threats posed by a compromised router: the attacker may attack by means of the routing protocol (for example, by sending false advertisements) or by having the router violate the routing decisions it should make based on its routing tables. The first kind of attack is often called an attack on the control plane, and the second is called an attack on the data plane. The first problem has received the lion's share of attention, perhaps due its potential for catastrophic effects. By contrast, the problem we are considering — subverting the forwarding process — has received comparatively little attention. This is surprising since, in many ways this kind of attack presents a wider set of opportunities to the attacker - not only denial-of-service, but also packet sniffing, modification and insertion - and is both trivial to implement (a few lines type into a command shell) and difficult to detect.

Related Work

The earliest work on fault-tolerant forwarding was done by Radia Perlman [11]. Several researchers have subsequently proposed lighter-weight protocols for actively probing the forwarding path to test for consistency with advertised routes. Subramanian et al's Listen protocol [13] does this by comparing TCP Data and Acknowledgment packets to provide evidence that a path is part of end-to-end connectivity, while Padmanabhan and Simon's Secure Trace route [10] achieves a similar goal using signed probe packets targeting intermediate routers. Both approaches only test for gross connectivity and cannot reveal whether packets have been diverted, modified, created, reordered or selectively dropped.

A very recent paper [3] presents a secure routing given the existance of a path of non-faulty routers between the source and the destination. The protocol is based on source routing, link by link message authentication, destination acknowledgment and timeouts. Later on, they found a vulnerability: A faulty router can make a correct router detect some other correct router as faulty. Their protocol can be fixed [2], but still it has a high overhead to be deployable in modern networks.

The approach most similar to our own is the WATCHERS protocol, which detects disruptive routers based on a distributed network monitoring approach [5, 4, 8] and a traffic invariant called conservation of flow. However, the WATCHERS protocol had many limitations in both its traffic validation mechanism and in its control protocol, many of which were documented by Hughes et al. [8].

Traffic Validation

A compromised router can make arbitrary alterations to the forwarding behavior of that router, but given the distributed nature of packet forwarding it is not possible in general for an adversary to perfectly conceal such behavior. As long as the packets traverse some uncompromised router, there is enough data redundancy to detect the alteration. Hence, implementing a traffic validation mechanism is an engineering problem.

The most precise way to validate traffic is store, at each router, a complete copy of the packets sent and the time at which each was forwarded. However, the storage requirements to buffer these packets and the bandwidth consumed by resending them make this approach impractical. In practice, designing a traffic validation function is a tradeoff between accuracy and overhead. In addition, real networks occasionally lose packets due to congestion, reorder packets due to internal multiplexing, and corrupt packets due to interface errors. Traffic validation needs to accommodate this abnormal but non-malicious behavior. That is, one must address an inherent tradeoff between an acceptable number of false positives and false negatives.

We have explored and implemented a variety of traffic validation mechanisms, including those based on approximate flow conservation, on comparisons between incremental hashes of packet content, and on set reconciliation protocols. While detailed descriptions and empirical comparisons of these approaches are outside the scope of this position paper, we have experience with several mechanisms that are hard to defeat, have few false positives and have acceptable implementation and state overheads.

Distributed Detection

The detection of a compromised router requires synchronizing the collection of traffic information and distributing the results for detection purposes. Since routers collect the information upon which traffic validation is based, there will be some uncertainty in determining which router is faulty. For example, suppose router r_1 collects traffic information about packets that traverse r_1, then a neighboring router r_2, and then a third router r_3. Based on the information r_3 has about the traffic it has seen and the traffic information r_1 has provided, r_3 can determine that packets has been dropped. But, r_3 can't determine whether r_1 is lying about what it claims to have forwarded to r_2 or whether r_2 has dropped the packets. Hence, there is an inherent lack of precision in determining which routers are compromised.

We have specified the problem of distributing traffic information and determining which routers are faulty. We cast the problem as a failure detector with *accuracy* and *completeness* properties. This failure detector reports suspicions as path segments, which are sequences of adjacent routers. More specifically, a failure detector reports a path segment π if it suspects a router in π is forwarding traffic along π in a faulty manner. A failure detector also has a precision, which is the maximum length of a path segment it suspects.

In terms of our specification, WATCHERS is accurate and has a precision of 1, but it is not complete. We have also developed two new failure detectors, both of which are accurate and complete. The two detectors differ in their precision: one has a precision of 2, and the other has a precision of $k+2$ where k is the maximum number of adjacent failure detectors that can be compromised. The less precise failure detector is significantly less expensive than the more precise one in terms of the amount of traffic information that needs to be maintained and in the amount of synchronization required among the routers collecting traffic information.

Response

Once a path segment π is detected as containing compromised routers, some countermeasure is needed to isolate the path from the routing fabric. An obvious countermeasure is to report the π to the administrator of the affected routers. In the meantime, the routers can update the forwarding tables to avoid π.

One question is how aggressive the countermeasure should be. If π is detected, then there is at least one router r in π that is compromised. An aggressive countermeasure would be to remove all of the routers in π from the routing fabric, on the theory that one should avoid using a router that has been compromised. Doing so, however, could have a serious impact on network performance. A less aggressive countermeasure would be to only remove the path segment π from the routing fabric. In doing so, r may still be routing packets, but not along π. Then, if r continues to behave in a faulty manner, then path segments containing r will continue to be suspected and removed which could lead to r becoming completely isolated.

Current Status

In this paper we have identified several tradeoffs in the design of fault-tolerant forwarding. We are exploring some of these tradeoffs through simulation and analysis based on synthetic and on actual topologies [12]. We are currently working on an implementation of the ideas described above. We are targeting a link-state protocol and are attempting to make as small of a change to the protocol and link update messages as possible. We are using the less precise failure detector because of its significantly lower overheard.

References

[1] Xuhui Ao. DIMACS Report: Workshop on Large Scale Internet Attacks, November 2003.

[2] Ioannis Avramopoulos, Hisashi Kobayashi, Randolph Wang, and Arvind Krishnamurthy. Amendment to: Highly secure and efficient routing, February 2004. Amendment.

[3] Ioannis Avramopoulos, Hisashi Kobayashi, Randolph Wang, and Arvind Krishnamurthy. Highly secure and efficient routing. In *Proceedings of INFOCOM 2004 Conference*, March 2004.

[4] Kirk A. Bradley, Steven Cheung, Nick Puketza, Biswanath Mukherjee, and Ronald A. Olsson. Detecting disruptive routers: A distributed network monitoring approach. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 115–124, May 1998.

[5] Steven Cheung and Karl Levitt. Protecting routing infrastructures from denial of service using cooperative intrusion detection. In *New Security Paradigms Workshop*, 1997.

[6] Gauis. Things to do in Ciscoland when you're dead, January 2000. www.phrack.org.

[7] Kevin J. Houle, George M. Weaver, Neil Long, and Rob Thomas. Trends in denial of service attack technology. CERT Coordination Center Technical Report, October 2001.

[8] John R. Hughes, Tuomas Aura, and Matt Bishop. Using conservation of flow as a security mechanism in network protocols. In *IEEE Symposium on Security and Privacy*, pages 132–131, 2000.

[9] Craig Labovitz, Abha Ahuja, and Michael Bailey. Shining light on dark address space, November 2001. Arbor Networks Tech. Rep.

[10] Venkata N. Padmanabhan and Daniel R. Simon. Secure traceroute to detect faulty or malicious routing. *SIGCOMM Computer Communications Review*, 33(1):77–82, 2003.

[11] Radia Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, MIT LCS TR-429, October 1988.

[12] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with Rocketfuel. In *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 133–145. ACM Press, 2002.

[13] Lakshminarayanan Subramanian, Volker Roth, Ion Stoica, Scott Shenker, and Randy Katz. Listen and Whisper: Security Mechanisms for BGP, NANOG 30, February 2004. http://www.merit.edu/ nanog/mtg-0402/subramanian.html.

[14] David Taylor. Using a compromised router to capture network traffic, July 2002. Unpublished Technical Report.

[15] Rob Thomas. ISP Security BOF, NANOG 28, June 2003. http://www.nanog.org/mtg-0306/pdf/thomas.pdf.